The geoSQL language for the manipulation of geographical data(*)

Franco Arcieri(1), Stefano Ercoli(1), Enrico Nardelli(2,3)

(1) Algotech s.r.l., Via Biella 10, 00182, Roma, Italy.
(2) Istituto di Analisi dei Sistemi ed Informatica, C.N.R., Viale Manzoni 30, 00185 Roma, Italy.
(3) Dip. di Matematica Pura ed Applicata, Univ. di L'Aquila, Via Vetoio, Loc. Coppito, 67100 L'Aquila, Italy, e-mail: nardelli@vxscaq.aquila.infn.it.

Abstract

In this paper some architectural issues related to the geographical information system CARTECH are presented and discussed. The system is based on an extension of the relational model. In this paper we focus our attention on the interactions among the different level of abstraction of the system. In particular we describe the relations between the logical data model and the underlying data structure used for its physical representation. The goal is the realization of a set of powerful and flexible operators for the management of geometrical-spatial data, in order to obtain a more efficient cooperation among the different modules of the system, accordingly to the adopted "system integration" approach.

1. Introduction

In the field of data base management system (DBMS) a growing interest is focused on the capability of efficently handling spatial-geometrical data [Sam91, SSD89, GA90, SSD91]. This arises from the need of managing, with DBMS technology new kinds of applications which are able to capture and to represent the reality of interest by means of models closer to user's view of real world. The difference with standard applications relies on the nature of the data and of the types used for their description and manipulation: now, in general, is not sufficient to efficently handle alphanumeric strings, but we want to represent the shape, i.e. the spatial location of the entity of interest.
It is easy to be aware that the complexity of this new kind of data implies significant efforts during the design phase at the different levels of abstraction for the definition of suitable logical data models capable of furnishing an uniform approach to the data, and of data structure that efficently maps models and allows the implementation of powerful and flexible operators.
A class of applications which play a key role in this context, is constitued by geographical information system or geographical data bases, as the efficient representation of the land and of the related elements, is becoming more and more relevant in many application fields, for instance - in the planning and design of services in the land, in the optimization of the use of natural resources, in the localization of geological sites, in the urban area administration, in map drawing, and so on. In the case of interaction with a system for the management of cartographic data, it is of particular relevance for the user the availability of specific operator of aggregation on spatial-geometrical data, the possibility of supporting direct spatial searches, based on geometric properties of object, and of supporting undirect spatial searches, which allow the identification of geometric entities on the base of their descriptive properties. Our approach for the implementation of a cartographic data management system, is based on the proposal presented in [GN90] which relies on a "system integration" methodology of different cooperating modules and where the integration is granted by an unifying logical data model.
In this paper architectural issues related to the geographical information system CARTECH are discussed. In particular discussion is focussed on the design of the basic operators for the management of spatial geometrical data that implement the logical data model operators. This has required the design of data structures able to support them efficiently and effectively.

This paper is structured as it follows: in section 2 a general description of our approach to the problem is presented and the architecture of CARTECH is sketched; section 3 presents geoSQL, our proposal of extension to the SQL language and examples of its use; in section 4 we describe the main features of GEOTECH, the library for the management of spatial data and finally, in section 5 query resolution strategies are discussed.

2. The architecture of CARTECH

In [GN90, GNT91a, AN92] an analysis of advantages and drawbacks of different approaches to the realization of geographical databases was presented. In the integration approach, it is necessary to have a logical data model the defines both a uniform reference schema for the specification of the integration between subsystems and provides the user with a high level data manipulation language. The model thus guarantee to the user a uniform and integrated view of geographical entities while allow an access trough both descriptive and spatial characteristics.

The idea on which we have based our proposal is that the

(*) Partially supported by research line "Multidata" of the "Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo"-CNR and working group "Basic GOODS" of the "ESPRIT - Basic Research Action"

descriptive component and the spatial-geometrical one of a given territorial datum have to be separated at the physical level, to obtain efficiency, but they have to be integrated at the logical level, so their representation and manipulation are simple and immediate. It has to be possible to refer to the shape of a region or to an entity that has a spatial-geometrical component, in the same way as to any traditional attribute of a relational DBMS. On the other hand, the treatment of spatial-geometrical data is fulfilled by special purpose functions assuring higher efficiency.

In [GNT91a] a model for the management of complex data based on an extension of the relational model was formally defined. In [GNT91b] its formal properties were investigated and its completeness and soundness were proved. The focus of the model is to introduce abstract data types for the specification of the attribute domains.

Our choice for supporting logical integration has been of relying on the well founded relational theory [Cod70], and to suitably extend the relational data model in order to represent in integrated way the descriptive and geometric part of geographical information. The possibility of treating in a relation geometric values as well as traditional alphanumeric attributes allows to extend the SQL language with new simple syntactic structures: topological predicates may be expressed on the geographical entities and it is possible to apply some geometric operators on these.

The SQL language has been enriched (from now on we will refer to this extension with the name of geoSQL) by means of the introduction of new relational operators based on functions for the composition of spatial-geometric components of the geographical entities, and for their aggregation, performed on the base of suitable conditions to be verified on the descriptive components of the entities. Since in geographical applications the area of interest is usually represented by thematic maps[1] the user has the possibility with these functionalities to easily create new maps by composing and aggregating existing ones according to its application needs. The new thematic maps are fully integrated in the system and can be further processed.

CARTECH architecture essentially consists of two functional modules: the relational DBMS and the spatial-geometric module, which implements a set of functions for the representation and processing of spatial-geometric data. The choice to represent and separately manage these two types of information, generates two different activities of data manipulation. The partial results have then to be compared and integrated for obtaining the final result. In particular, an interpreter checks the syntactic correctness of the issued queries and calls the modules appropriate to their resolution. The strategy for queries resolution firstly manages the geometric part of the query and then makes use of the outcome for the resolution of the relational part by using the relational DBMS. The different actions carried on by the relational DBMS and by the spatial module are completely hidden to the user. He can access and manipulate data only through the primitives of the geoSQL language.

[1]A thematic map is a geometrical subdivision of a piece of land in areas homogeneous with respect to values of a set of attributes, together with a table which associate to each homogeneous area the given set of values for the attributes .

The above described approach leads to an architecture, shown in figure 1, consisting of the following functional modules:

- an *interface* which allows the users to access to system;
- a *preprocessor* for the identification and separation of the descriptive and geometric parts of the query;
- a *query processor* which coordinates the different phases of query resolution and checks the consistency of the cross-references among the geometric components, contained in the library of geometrical data, and the keys-names managed in the relational DBMS
- a *processor of geometric queries*; which manages geometric subqueries and creates temporary tables containing the partial results obtained from the their resolution;
- a *relational DBMS* that resolves the descriptive part of the query using the temporary tables produced by the spatial query processor.

The *interface* connects the system to the human user or the application programmers: it makes available the system functionalities, recognizes the commands and displays the result of the operations.

The *preprocessor* divides spatial subexpressions, involving exclusively geometric attributes, from non spatial ones, which refer to alphanumeric attributes. Spatial subexpressions can be found either in *select* clauses as geometric operators, or in *where* clauses as topological predicates.

The preprocessor is also recognizes the new aggregation operators introduced in geoSQL. This functionality is carried out by a syntactic analyzer that sends the identified subexpressions to the geometric query processor for their evaluation.

The *query processor* drives the query resolution and uses a system relational table, namely "geometry", in order to mantain the relationships between the user's names and those internal to the geometric data structure.

The *relational DBMS*, based on the INGRES system, evaluates the queries concerning alphanumeric data and executes the commands for creating and modifying the temporary tables used by the spatial processor.

The *geometric query resolutor*, based on GEOTECH library, accepts as input spatial subexpressions. It carries out the computation of the temporary tables containing partial results, that is tables containing the regions satisfying the imposed topological conditions. Also, it computes the required geometric functions. This activity produces geometric objects in the case of union or intersection operators, numeric values when the area of the specified regions is computed.

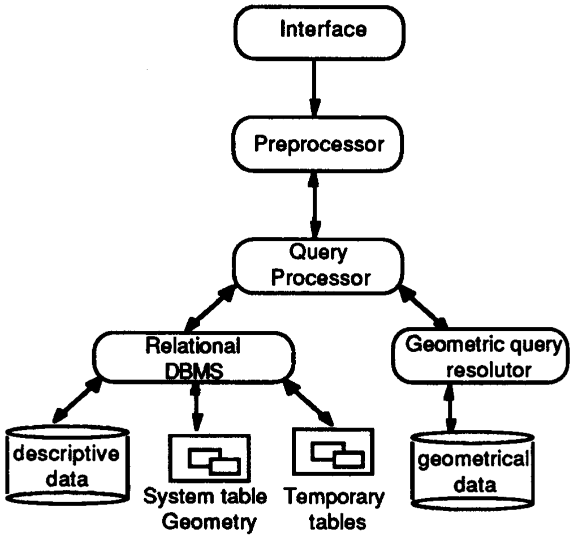

Fig. 1: CARTECH System Architecture

3. The geoSQL language

The geoSQL language [AN92] extends SQL to deal with the new relational operators, G-Compose and G-Decompose, and with the ADT Geometry(S).
In the geoSQL language are implemented the operators COMPOSE and MERGE for the G-Compose and the operator SEPARE for the G-Decompose.

The COMPOSE operator groups the values of one or more geometric attributes applying to the grouped values the fusion fuction ∪ and discards the remaining attributes. The grouping is done on the basis of the equality of others attributes of the relation.

The MERGE operator groups the values of one or more geometric attributes applying to the grouped values the fusion fuction *geo*.

Note the different effect of the application of the two operators: in the first case regions with more than one component may be given as output results, in the second one only regions with one component are produced. It follows, in the last case, that the resulting relation cannot be disaggregated in the original components.

The possibility of using geometric conditions in the *where* clause makes it possible to select, with more precision, tuples on which apply the aggregation operators.
Notice that a safe use of the fusion fuctions in the COMPOSE and MERGE operators is assured because, due to ADT approach, the manipolation fuctions allowed on set of values of an attributes A_i are only the operation defined in the ADT specification of *type*(A_i).

The SEPARE operator perform the inverse function of the COMPOSE one. Each Shape of the geometric attribute on which the SEPARE is applied, is decomposed in its elementary components. Such an operation can be considered as a normalization action on the relation.

We give in the following two simple examples in order to show the typical syntax of the COMPOSE and MERGE operators.

Assume the relation MINERALS is given accordingly to the table of figure 2, and that one wants to aggregate the area of exploitation of each mineral on the base of its family.

The following geoSQL expression can be issued:
COMPOSE minerals.Shape
BY minerals.Family FROM minerals;

Family	Type	Shape
slate	type_11	{ {min_11} }
slate	type_12	{ {min_12} }
sandstone	type_21	{ {min_21} }
sandstone	type_22	{ {min_22} }
clay	type_3	{ {min_3} }
limestone	type_41	{ {min_41} }
limestone	type_42	{ {min_42} }
limestone	type_43	{ {min_43} }

Figure 2 : relation MINERALS

The operator group the values of the attributes minerals.Shape applying the fusion fuction ∪ on the basis of the equality of the values of the attribute minerals.Family. Relation FAMILY_OF_MINERALS, in figure 3 below, shows the result:

Family	Shape
slate	{ {min_11} , {min_12} }
sandstone	{ {min_21} , {min_22} }
clay	{ {min_3} }
limestone	{ {min_41} , {min_42} , {min_43} }

Figure 3 : relation FAMILY_OF_MINERALS

The application of the operator MERGE would have instead produced the following table (figure 4):

Family	Shape
slate	{ {min_11, min_12} }
sandstone	{ {min_21, min_22} }
clay	{ {min_3} }
limestone	{ {min_41, min_42, min_43} }

Figure 4 : relation FAMILY_OF_MINERALS_MERGED

Note that here each entity has a geometric value which is a Shape with only one component.

The sintax of the introduced operators is , in the general case:

COMPOSE x BY y FROM r WHERE p;
MERGE x BY y FROM r WHERE P;
SEPARE x BY y FROM r;

To cope with the manipulation of ADT Geometry(S) a number of topological predicates and geometric operators have been introduced.

The SQL clauses SELECT and WHERE in the Geo SQL supports in a uniform manner both the traditional and Geometry-valued attributes; in particular in the clause SELECT it is now possible to apply the introduced geometric operators and within the WHERE clause it is possiblle to express predicates with geometric attributes.

Geometric operators are the implementation of the corrispondent operators introduced in the Geometry(S) definition. In particular we have implemented the three binary operators which give a new region:

- INTERSECTION: applied to SHAPES A and B gives SHAPE intersection between A and B, that is the SHAPE whose components are given by the equality of a component of A with a component of B.
- INTERSECTION*: applied to SHAPES A and B gives the SHAPE whose components are the results of the intersection of every component of A with every componet of B.
- UNION: applied to the SHAPE A and B furnishes the SHAPE C whose components are components of B or A.

A unary operator it is also introduced:
- AREA: it calculates the area of specified region surface.

The topological predicates provide a boolean result. In all cases the arguments are two geometric attributes, which may possibly belong to the same relation.

- INTERSECT: it returns "true" when the regions represented by the two attributes have a non empty intersection; "false" if the two regions have an empty intersection.
- INTERSECT*:it returns "true" when almost one component of region A intersect one compont of region B ,"false" otherwise.
- INCLUDE: it returns "true" if the region represented by the second attribute is included in the first one; "false" otherwise.
- ADJACENT: it returns "true" if the two regions are adjacent; "false" if they are not.
- EQUAL: it recognizes when two regions are equal. That is, it returns "true" if the regions represented by the two attributes have the same shape and location in the plane; "false" otherwise.

We present now an example of application to Land Resource Management. We suppose the database schema is composed by two relations, the former describes the presence of minerals on the land, the latter describes the constraints presents on the land itself:

MINERALS_BY_QUANTITY(name,
exploitation_quanti-ty, shape);
CONSTRAINTS(type, shape);

During the interaction the user may be interested into the definition of exploitation plans by means of the construction of particular thematics maps, called exploitation maps, which describes the regions where the materials can be exploited, taking into account the constraints imposed on the land. Therefore it is interesting to evaluate which constraints can be relaxed in order to reach the objective of collecting a predefined amount of a given material. A possible methodological approach is described in what follows.:

We start building a new relation, MINERALS, which groups the exploitation lands by type of mineral:

```
    COMPOSE minerals_by_quantity.shape
        BY minerals_by_quantity.name
        FROM minerals_by_quantity;
```

The schema of the obtained relation is:
MINERALS(name, shape);

From the relation MINERALS and CONSTRAINTS, we can find the intersection between the pieces of land subject to constraints and the lands where there are minerals by issuing:

```
SELECT minerals.name, constraints.type
INTERSECT*(minerals.shape, constraints.shape) shape
    FROM MINERALS, CONSTRAINTS
    WHERE
        INTERSECT*(mineral.shape,constraints.shape);
```

The new relation MINERALS_BY_CONSTRAINTS with the schema
MINERALS_BY_CONSTRAINTS(name, type, shape);
is obtained, where the istances of the new geometric attribute 'shape' are the intersection between each land where is a mineral and each land subject to a constraint.

It is possible built other news maps to group all the lands where we can exploit a specific mineral and where the amount we can obtain is greater than a fixed threshold (say 500):

```
COMPOSE minerals_by_quantity.shape
    BY minerals_by_quantity.name
    FROM minerals_by_quantity
    WHERE (minerals_by_quantity.name = 'clay')
        AND (mineral_by_quantity.exploitation_quantity >
500);
```

4. Implementation of the ADT *Geometry*(S)

The goal of realizing a logical data structure which reflects the way the user looks at the data requires the support of non-atomic domains, capable of represent in a uniform way the spatial-geometrical component of an object. The above presented model permits the integrated representation of geographical entity, by means of the introduction of the abstract data type (ADT) Geometry(S), which supports SHAPE-valued attributes. A single element of the SHAPE domain, instance of the ADT Geometry(S), univocally defines the shape and the position of an object, hence the spatial-geometrical nature of an object can fully be represented at logical level by the value of a single attribute. The user way of looking at geographical entities is therefore fully supported.

To implement such a model we have chosen to represent the spatial geometrical nature of the objects by a discrete set of points in the plane. The set S, which represents the set of the atomic elements on which is based the ADT Geometry(S), has been defined as a finite subset of the integer coordinates points of the plane(raster plane). One element with shape value will be therefore composed by one or more elementary components each one of which can be described by a finite subset of plane points as well ($H_S = P(P(S))$).
The implementation of ADT associates to each region of the plane one SHAPE valued element, and as consequence, a region is viewed as a finite subset of elementary components.

In order to represent such elementary components a data structure, namely Multivalued Quadtrees, has been used. With such a structure we do not need to esplicitely represent all the couples of integer which belong to the component, but for each component only a limited amount of information is memorized, although each single point can be reached by means of the appropriate algorithm.

The GEOTECH library employs multivalued quad-trees [ArD89, AENP93], a direct extension of quad-trees [Sam84]. A multivalued quad-tree is a hierarchical data structure, introduced and analyzed in [ArD89] and refined in [ABN91], for the representation of sets of regions overlapping on the plane. An example of a multivalued quad-tree is shown in figure 5. In the normal quadtree only regions which do not overlap are allowed. It is built starting from a regular decomposition of the plane in quadrants. The plane on which the queries are defined is represented as an array of $2^n x 2^n$ elements. It is recursively split in quadrants and subquadrants until we arrive either at empty quadrants or at quadrants which entirely belong to a (set of) region(s).

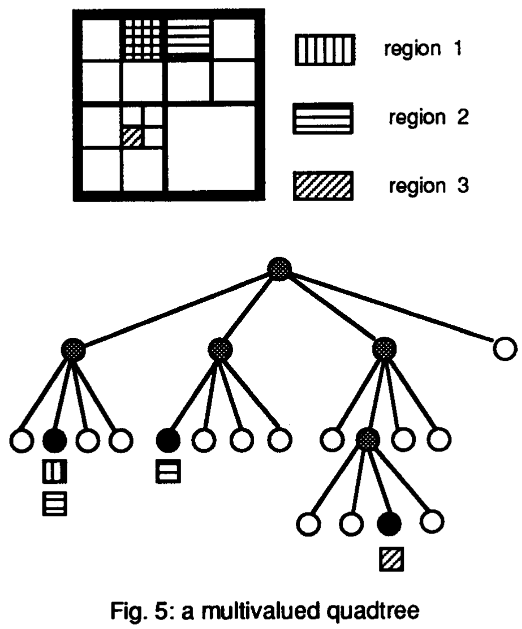

Fig. 5: a multivalued quadtree

This decomposition may be represented with a tree, in which every internal node has four children. The leaves of this tree corresponding to homogeneous decomposition blocks are black: to allow the representation of overlapping regions, to every black node is associated a pointer to the list of all regions which share the block. Leaves corresponding to empty areas are marked white. All the other nodes, which are internal nodes, are marked gray.

The simplest implementation of a quad tree is by means of a pointer data structure. This solution has the drawback since it requires a big quantity of memory.
For this reason the implementation has been based on a linearized view of the quadtree built by means of a depth first order visit of the quad tree and storing the order on which nodes are visited in an array. Such a structure, called linear multivalued quad-tree, has the advantage of resulting very compact and non restrictive, as any operation which can be realised on the tree can be realised on the linear data strucure [AENP93]

The correspondance among elementary components stored in the multivalued quad tree and the regions it is realized by means of a structure which furnishes the mechanism for associating to each component the region it belongs to. The strongest requirement of this association is the efficiency. The adopted solution uses arrays whose elements contains the references to regions. The array indexes, which are the identifiers of the elementary components, realizes the association mechanism from the components to the belonging regions. A simple direct acces to an array element establishs the link between elementary component and its region. Moreover, to each multivalued quad tree we can associate more regions arrays, each one of which can be considered as a particular view on the same geometry.

Such a structure allows to realize an efficient implementation of the composition functions, which characterize the new introduced operator of COMPOSE and MERGE In particular a COMPOSE operation on a geometric attribute is simply realised by building a new view of the same geometry, that is by defining of a new array of the regions to be associated to the same quad tree, where the elementary components that constitutes the regions are represented.

6. Query resolution

Before describing the technique adopted for the query resolution, a premise is needed about the way used to identify the regions in the database. The links between spatial data and descriptive data are maintained by means of numeric identifiers; in the geometric attributes of a relational table, the region is indicated by an integer, that identifies the region among those stored in the same array of the regions.

In the evaluation of a query, the system firstly verifies the syntactic correctness of the input string. Secondly, SELECT clause is processed. Whenever one geometric predicate is recognized, the geometric query processor manages it and inserts the results in a temporary relational table. After the processing of all geometric predicates, the query is slightly modified: firstly, the just solved predicates are deleted from the WHERE subclause and suitable constraints are added to impose that resulting tuples belong to these temporary tables; then we temporarily eliminate the references to geometric operators in the SELECT clause and leave only their operands. This modified query is ready for being processed by the relational DBMS, which stores the results of this evaluation in a new relation. This is given back to geometric query processor which can now process the geometric operators. Finally, the system presents in output the requested data.

If clauses COMPOSE or MERGE are recognized, predicates which aggregates geometrical data are processed first; the result is inserted in a table whose schema contains also the columns for the new geometric attributes and whose regions will be given by the application of geometric functions. Then a new temporary table is built whose schema is given by the columns on which the composition functions operate together with the columns for the new geometric attributes. Such a table contains all the information for constructing new vectors of the regions associated to geometrical attributes. The schema of the final relational table is built accordingly to the requirements given by the user in the particular command.

We can clarify at this point, how the system keeps the consistency of the data. All the regions belonging to a geometric attribute of a given relation are stored in the same array of regions. The name of the file containing the array of the regions is univocally associated to the couple relation-name, column-name, such an association is transparent to the user and is managed completely by the system by means of system-defined relational table which mantains also the correspondence between array of the regions and the underlying geometry, that is with the file that contains the multivalued quadtree. As already mentioned, different arrays of regions can refer to the same quad tree.

We have illustrated a possible strategy for query resolution. Another possibility is to firstly calculate the relational side of the query and to next use these results for evaluating the spatial side. The choice between the two possibilities can only be made on the basis of considerations depending on the specific instance of the query and on the current extension of the database, which are usually not known a priori. It is reasonable to choose the strategy that minimize the amount of the data to process: but it is not possible to specify a general strategy which is always valid. A further possibility of optimization is to provide an information exchange between the relational DBMS and the spatial processor for decreasing the complexity of the query processing algorithms involved. To this end, an approach which is currently under investigation, is the use of the so-called "on-line" approach to quickly compute partial approximate solutions to spatial queries [FN91] and to use this information to drive the query optimization process.

5. Conclusions

In this paper we have presented and discussed a system for geographical data management, CARTECH. The system is based on a logical data model developed as extension of the relational model. It also features an interface based on an extension of SQL to deal in a uniform way with descriptive and spatial-geometric information, which is the specific focus of this paper. CARTECH has been implemented and runs on 80386 machine, under the SCO Unix operating system and interfacing with INGRES database management system.

The usability analysis of the applications developed using CARTECH are encouraging both for the manageability and simplicity of use, and for the efficiency in the treatment of geographic queries. The geoSQL language, introduced for the geographical data manipulation, turns out to be a simple and powerful instrument, being able to satisfy the demands of different application problems. The behaviour of the data structures adopted for the resolution of topological queries seems really good, as it results from the efficiency comparison between our system and the commercial systems with analogous functions and comparable computing power.

The extreme modularity of the system architecture allows to easily extend the data types that it can manage, keeping always separated the manipulation of the descriptive component from the more complex ones (i.e.: three-dimensional data).

Acknowledgments

We are greatly indebted to Paolo Dell'Olmo, Giorgio Gambosi, Marinella Gargano, and Maurizio Talamo for many fruitful discussions on these issues and their contributions to the work here described. Many thanks also to Luigi Barella for his contribution to system design and implementation.

References

[AENP93] F. Arcieri, S.Ercoli, E.Nardelli, G.Proietti: Multivalued quadtrees for the efficient processing of spatial data, manuscript, 1993

[AG90] F. Arcieri, G.Gambosi: Ambienti innovativi per la realizzazione di applicazioni di supporto alla pianificazione di insediamenti sul territorio, Conference FAST, Venice, February1990

[AGLN90] F. Arcieri, G.Gambosi, M.Lancia, E.Nardelli: Un sistema per la gestione di dati spaziali e di tematismi definiti sul territorio, Conference of the Italian Association for Automated Calcoulous (AICA), Bari, October 1990

[AN92] F. Arcieri, E.Nardelli: An integration approach to the management of geographical information: CARTECH, 2nd Int. Conf. on Systems Integration, Morristown, NJ, June 1992.

[ArD89] F. Arcieri, P. Dell'Olmo: A Data Structure for the Efficient Treatment if Topological Join, Fourth International Symposium on Computer and Information Sciences, Turkey 1989

[ArTa90] F. Arcieri, M. Talamo: T.M.M.S.: Un sistema informativo territoriale per lagestione di dati spaziale e di tematismi definiti sul territorio: un'applicazione al Piano Regionale delle Attività Estrattive nella Regione Lazio, Proc. of 1990 Conference on State of the Art in Territorial Information Systems, Italian Chapter of AM/FM GIS, Rome, December 1990

[Cod70] E. F. Codd: A relational model for large shared data banks, Comm ACM vol.13, No. 6, June1970

[FN91] P.G. Franciosa, E. Nardelli: On-line approximation of quadtree border, Int. Workshop on DBMSs for geographical applications, Capri, May 1991.

[GA90] Proceedings of First International Workshop on DBMSs for geographical applications, Capri, Italy, May 1991.

[GN90] M. Gargano, E. Nardelli: A logical data model for integrated geographical databases, Proc. of 1st International Conference on Systems Integration, Morristown, NJ, April 1990.

[GNT91a] M. Gargano, E. Nardelli, M. Talamo: Abstract Data Types for the logical modeling of complex data, Information Systems, Vol. 16, N. 6, 1991.

[GNT91b] M. Gargano, E. Nardelli, M. Talamo: A model for complex data: completeness and soundness properties, Int. Workshop on DBMSs for geographical applications, Capri, May 1991.

[Sam84] H. Samet: The quadtree and other related hierarchical data structures, Computing Surveys, vol. 16, no. 2, June 1984

[Sam91] H.Samet: Spatial database system based on SQL, Proc. VLDB 91, June 1991.

[SSD89] Proceedings of First Symposium on Design and Implementation of Large Spatial Databases, Santa Barbara, Ca., July 1989.

[SSD91] Proceedings of Second Symposium on Large Spatial Databases, Zurich, Switzerland, August 1991.